Section 4 – Main Flow of the Algorithm in query.js

This section describes the main flow of the algorithm that handles queries in the system. The process consists of six main steps, ensuring that the query is processed efficiently and accurately.

Step 1: Question Refinement

The system receives the user's original query (originalQuery) and subjects it to a refinement process to improve its clarity and precision.

The polishQuestion() function is called, which delivers a **perfect prompt** to continue the execution of the algorithm in the following steps, without the need to maintain history for each next step.

- The getHistory() function retrieves the previous queries' history.
- A SmartFunction is used to restructure the query, considering the allowedContext.
- Metrics are recorded for input and output tokens.

Step 2: Identification of Relevant Files

With the optimized query, the algorithm searches for the most relevant files in the document index.

The SmartFunction identifyFiles() is called. Its purpose is to have the AI itself determine, given the query and the chunk index: "**Which chunks would you go to in order to find this information?**"

- The document index available in _Document_Index.txt is accessed.
- Files containing relevant information for the query are filtered.
- The number of files to evaluate is limited using the MAX_CHUNKS parameter.
- Token usage metrics are updated.

Step 3: Building the Answer

The system searches for answers to the query within each of the chunks listed in the previous step. The original question is combined with the refined prompt (the perfect version) to enhance the search.

- Key information is extracted from the relevant documents.
- MAX_ANSWERS is used to limit the number of answers retrieved.
- The tokens used in this phase are recorded.

The result of this step is a string containing the text of one or several valid answers, which will be polished in the next step.

Step 4: Answer Polishing

If relevant answers are found, the system refines the output before displaying it to the user.

- A SmartFunction is used to enhance the coherence and readability of the finalAnswer..

- Markdown formatting is applied, highlighting key ideas with bold text and bullet points.
- Tokens consumed during this step are counted again.

Step 5: History Update

The system saves the query and its response in the history to improve future interactions.

- The polished query and its answer are stored in the database.
- This history allows the system to maintain context during extended sessions.
- The number of iterations stored in history depends on the MAX_HISTORY setting found in the ".env" file.

Step 6: Cost Calculation and Response Delivery

Finally, the computational cost is calculated and the response is sent to the user.

- The input and output tokens used throughout all steps of the query are measured.
- The total execution cost is estimated.
- The polished answer is returned to the user in JSON format.

With this structure, the system ensures efficiency, accuracy, and optimization for each processed query.

Below is a screenshot of the six steps of the algorithm in action: